CS 6905 – Fall 2017

Software Engineering

Assignment # 2 - Online Voting System

Submitted by:

Group: # 4

Aafia Jabeen – 201790648

Aswini Naveen – 201692818

Haris Khan – 201454147

Due Date:

20 October 2017

Contract

Software Development Agreement

Our team has been contracted to design a system to develop an online voting system that will provide support for election officials to conduct an election.

Requirements

The vision of this project is to design generic **Online-Voting** system that will provide support for election officials to conduct an election (e.g. parliamentary, student union, sport club). This system design support popular election systems primarily winner takes all or proportional. If in case election officials decide to choose proportional election model then there is a threshold percentage votes to be achieved for a party to gain membership in a parliament. More than a one round of elections is possible. Additionally, the system shall provide directions to voting stations. For winner takes all, the election official can also select if the voter can select preference for candidates. After election the least popular candidate is dropped and votes re-assigned according to second choices, and so on, until one candidate has a majority of ballots. The election official will also be able to select districts participating in election and will be able to update the candidates.

We have agreed that the program will be developed in NodeJS, the frontend in Angular and database would be MongoDB.

Developers

Aafia Jabeen

Ashwini Iyer

Haris Khan

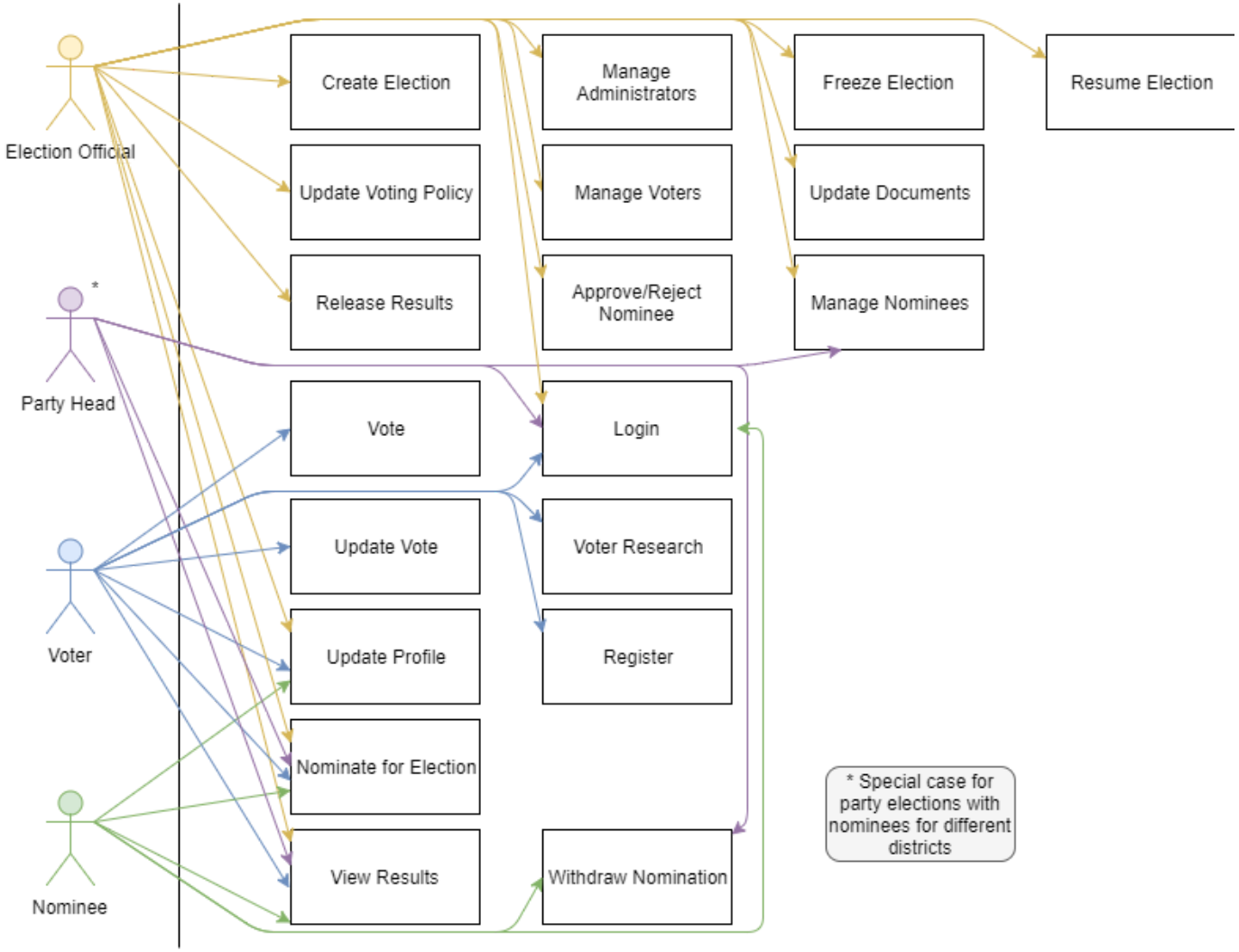
Client

Dr. Adrian Fiech

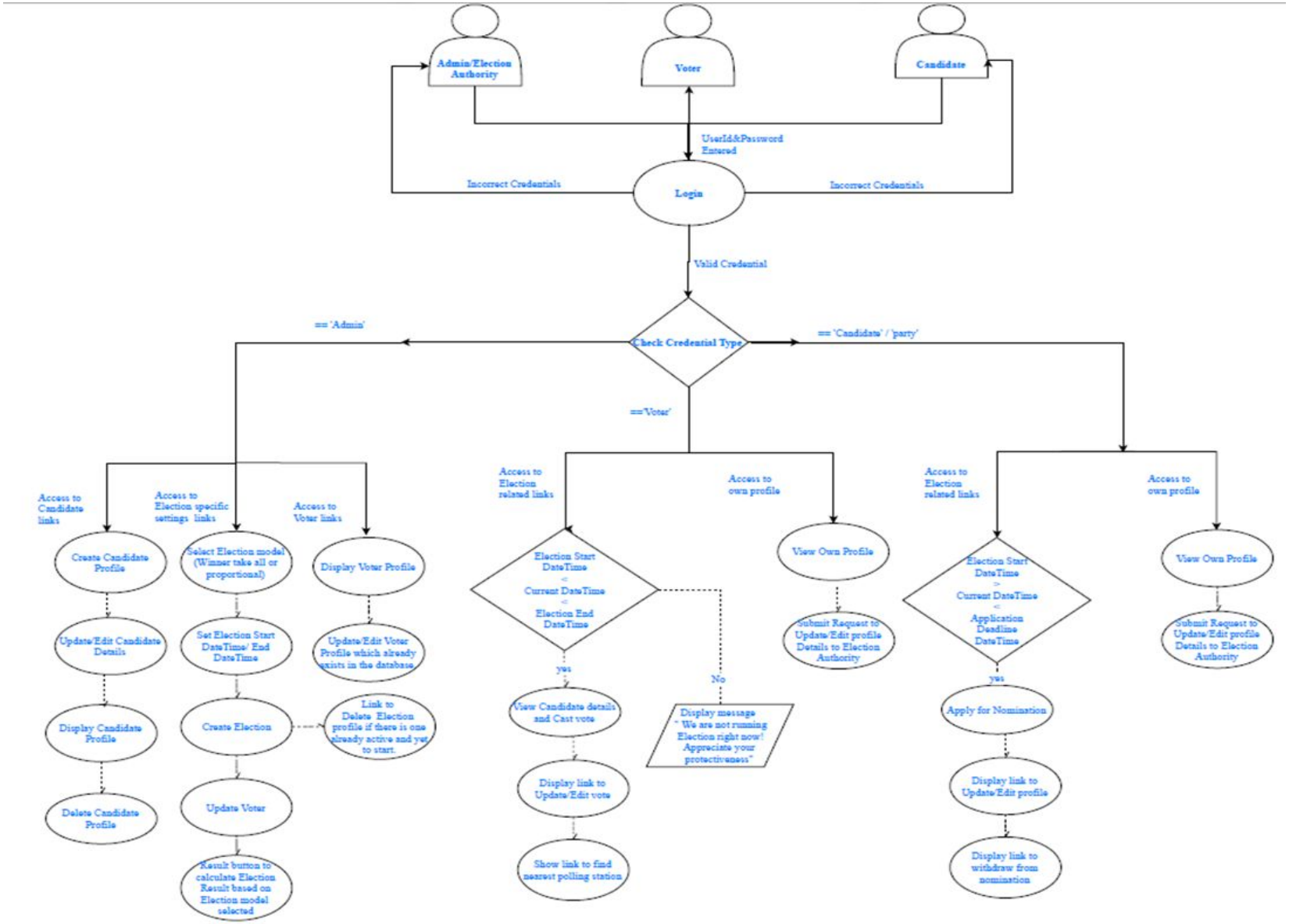
Use Case Diagram

Use Cases Described Below:

1. Create election
2. Vote
3. Nominate for Election
4. Voter Research
5. Release results
6. Update Vote
7. Freeze Election
8. Update Documents
9. Update Voting Policy
10. Nominee Update Profile



Alternate Use Case Diagram – System Level (Basic)



Use Cases

Use Case # 1 - Create Election

Name:

Create Election

Description:

This use case will allow the election official (organizer) to conduct an election. The election official will create the kind of election to be conducted (eg: parliamentary, student union, etc) and will also update the details of various districts (where the elections to be held) and the nominees. Upon request from the nominee or the voter, the use case will allow the election official to make changes in the personal details of the nominee or voter after proper verification. The election official prepare electoral roles and will update the voter's list from time to time. The schedule of election for filing the nominations are also issued. The election official will set up the rules for voting like the eligibility criteria, documents to be submitted and the verification required to apply for nomination, time period for a particular election and the number of rounds to be conducted. It lays down guidelines for the conduct of political parties and candidates during an election period. After the elections are conducted the election official will also be responsible for the release of results.

Actor:

- Election official

Basic Flow:

1. Election official signs into their account.
2. Selects create election
3. Update the election settings for the kind of elections to be held
4. Set the time for when the election starts
5. Set the time for when the election ends
6. Select winning strategy (e.g. winner takes all, proportional)
7. Select number of districts
8. Select predefined candidates or nomination based
9. Select the users who can vote (database, csv file, regular expression)
10. Add conditions for the user object that can vote (e.g. if only graduate students can vote or people can vote for their district only)
11. Select create

Alternate Flow:

6.

a) If (winner takes all) then select single vote or preferential vote

8.

a) If predefined selected then add candidates for districts
b) If nomination selected then
    i) Select either nominations has to be approved by an official or not
    ii) Add regular expression for users who can nominate (optional)

9.

a) If database then add the credentials and API URL for the database and the query for the database
b) If csv file then either add the URL and credentials to access it or upload the file
c) If regular expression then add the regular expression for the email address

11.

a) Select save as draft.

Preconditions:

- Election official is registered
- Election official is verified

Postconditions:

- Election to be held is created (if not saved as draft)
- Personal details of the nominee/voter are updated (if applicable)
- Cancels the nomination for a nominee (if applicable)

Use Case # 2 - Vote

Name:

Vote

Description:

The purpose of this use case is to allow a voter to cast a vote. The voter will have privileges like casting a vote, modifying it later and also to view the results. All eligible voters (residents, students, specific group of people) will be pre-registered and given access to the application. Voter can change personal details which will be validated by the system and stored in the database.

Actor:

- Voter

Basic Flow:

1. Voter signs into their account.
2. Selects vote to cast a vote
3. View for which candidate he has voted.
4. Selects update vote (if applicable).
5. View the results after the elections are over.

Alternate Flow:

1. Voter signs into their account.
2. Cannot cast a vote (deadline has passed for voting)
3. Voter is notified by the system
4. View the results

Preconditions:

- Voter is registered

Postconditions:

- Able to cast a vote
- Voting results can be displayed
- Personal details are updated (if applicable)

Use Case # 3 - Nominate for Election

Name:

Nominate for Election

Description:

This use case goes through the process of self nomination. If in case the candidate is new he /she can register their own profile and get an username/password assigned. If the candidate belongs to a specific party provided the party or the candidate already has existing account then they can login to the account and submit a new nominee application or update their existing profile (if required) or withdraw from nomination.

Actor:

- Nominee

Basic Flow:

1. User signs into their account.
2. Selects the election.
3. Selects nominate.
4. Selects self nominate or nominate somebody else.
5. Enter the details for the nominee.
6. Save

Preconditions:

- User is registered.

Postconditions:

- New nominees are sent for approval or are registered depending on voting policy

Use Case # 4 - Voter Research

Name:

Voter Research

Description:

This use case will allow the voter to perform some research on the website by selecting a research task from the list. A voter can select the category of election and the type of information he is looking for: election laws, nominees list, the policies and objectives of the political parties , etc. The voter is then presented with the appropriate type of documents by the system stored in the database. He can choose to view any number of required documents from a menu of options for the selected category of research. A voter can also search for the document he is looking for through a 'search' tab. The system will extract the required information from the database.

Actor:

- Voter

Basic Flow:

1. Voter signs into their account.
2. Selects the category of election for which the information is required
3. Selects the type of information from the list
4. Selects the appropriate document
5. Selects 'stop viewing' if he wants to stop looking for information.
6. Voter is asked if he needs any other information from the system and if he got all the information he was looking for
7. If yes, voter will be asked if he would like to perform any other task in the session
8. If no, voter will be redirected to the research task list.

Use Case # 5 - Release Results

Name:

Release results

Description:

This use case declares the winning candidate or the political party after the elections. It will determine the number of political parties or independent candidates participated in the elections, count the number of votes for each candidate or political party, decide the candidate with the maximum number of votes and declare the winner. It will also determine the voting difference for which a particular candidate has won the election and release the result along with the winning candidate details.

Actor:

- Election official
- Administrator

Basic Flow:

1. Election official/Administrator signs into their account.
2. Select the election.
3. Selects the category of election.
4. Checks the number of political parties / candidates participated in the election.
5. View the candidate with the maximum number of votes.
6. Release the results for winning candidate or political party along with candidate details.

Use Case # 6 - Update Vote

Name:
Update Vote

Description:

This use case allows users to change vote if they had voted already. Once the user is logged in the users are presented with list of election's those are created and that those they are eligible to vote. If the user selects the election that he has not voted and the deadline to vote is still open then he gets redirected to cast vote page with a message indicating he will be eligible to change after he/she had completed voting. If the user selects the election that he had already voted and deadline to vote has already passed then then he will be displayed with the details of his choice. If the user selects the election that he had already voted and the deadline to vote is still open then he is he will be eligible to change his vote.

Actor:
Voter

Basic Flow:

1. Voter is Registered and logged in.
2. Provided with list of Elections conducted so far and that he/she was eligible to vote.
3. Chooses the election to change vote.
4. After changing, selects Yes or No for final confirmation.
5. If Yes selected the latest changes are saved.
6. If No, then old value is retained.

Use Case # 7 - Freeze Election

Name:

Freeze Election

Description:

The Election Official or the administrator will be able to freeze the election so that nobody can vote. This would be a special case scenario when the election has to be halted, it can be resumed later on.

Actor:

- Election official
- Administrator

Basic Flow:

1. User is authenticated as election official or administrator
2. User selects the election to be frozen.
3. User selects freeze.
4. User authenticates again.

Use Case # 8 - Update Documents

Name:

Update Documents

Description:

The Election Official or the Administrator have the privilege to edit/update the document/profile of candidate or the voter.

Actor:

- Election Official
- Administrator

Basic Flow:

1. The Election Official or the Administrator is signed in.
2. Selects Update Documents.
3. Choose type of user Voter or Candidate.
4. Provides the unique UserId (either Voter_id/Candidate_Id/*Party_Id).
5. Clicks search to locate the profile.
6. Once profile is loaded edits the document.
7. Clicks on Save changes.

Use Case # 9 - Update Voting Policy

Name:
Update Voting Policy

Description:

This use case allows the Election official or the administrator to update the voting policy for a particular election. The voting policy will be set by the election official at the time of creating the election and can be updated later if required. In order to make changes in the policy, the admin/election official is required to select the category of election for which the voting policy needs to be updated. The admin/election official is required to enter valid input data example date and time in the system. If the data is invalid, he will be notified by the system through an error message and will be asked to re-enter the information or cancel the task. After the voting policies are updated, submit can be used to save the data.

Actors

- Admin
- Election Official

Basic Flow:

1. Admin is logged in.
2. Selects the category of election for which the voting policy needs to be updated.
3. Make the changes. Enter the valid data.
4. In order to save the data, click on submit.
5. The admin will be asked to confirm the changes made.
6. If yes is selected, the changes are saved and the voting policy is updated.

Alternate Flow:

1. Admin is logged in.
2. Selects the category of election for which the voting policy needs to be updated.
3. Make the changes. Enter the invalid date.
4. An error message generates and admin is asked to enter the valid date or cancel the task.
5. Admin cancels the task and close the session.

Preconditions:

- Admin is registered
- Admin is verified

- The voting policy is already set for that particular election by the election official

Postcondition:

- The voting policy is updated.

Use Case # 10 - Nominee Update Profile

Name:
UpdateProfile - Nominee

Description:

This use case enables the registered nominees to make changes in their personal information by providing the appropriate data with the supported documents. The nominee selects the type of information he needs to update example: personal information, background details. After the data is entered, the nominee will be asked to upload the supported documents. The documents will then be verified by the election official before the final changes reflects in the system. Till the election official approves the changes, the request will be in pending state. After making the changes and uploading the documents, the nominee can save the changes and close the session.

Actors

- Nominee

Basic Flow:

1. Nominee is logged in.
2. Selects the category of information he would like to update.
3. Enter the required and valid data.
4. Upload the supported documents.
5. Save the changes through the submit option.
6. Nominee will be asked by the system to confirm the changes made.
7. Select yes and close the session.

Alternate Flow:

1. Nominee is logged in.
2. Selects the category of information he would like to update.
3. Enter the required and valid data.
4. Upload the supported documents
5. The task failed as the system fails to accept the uploaded document.
6. Nominee is asked to upload the document again.
7. The task failed again and after a number of failed attempts the task is cancelled.
8. Nominee is asked if he likes to perform another task.
9. Selects No and close the session.

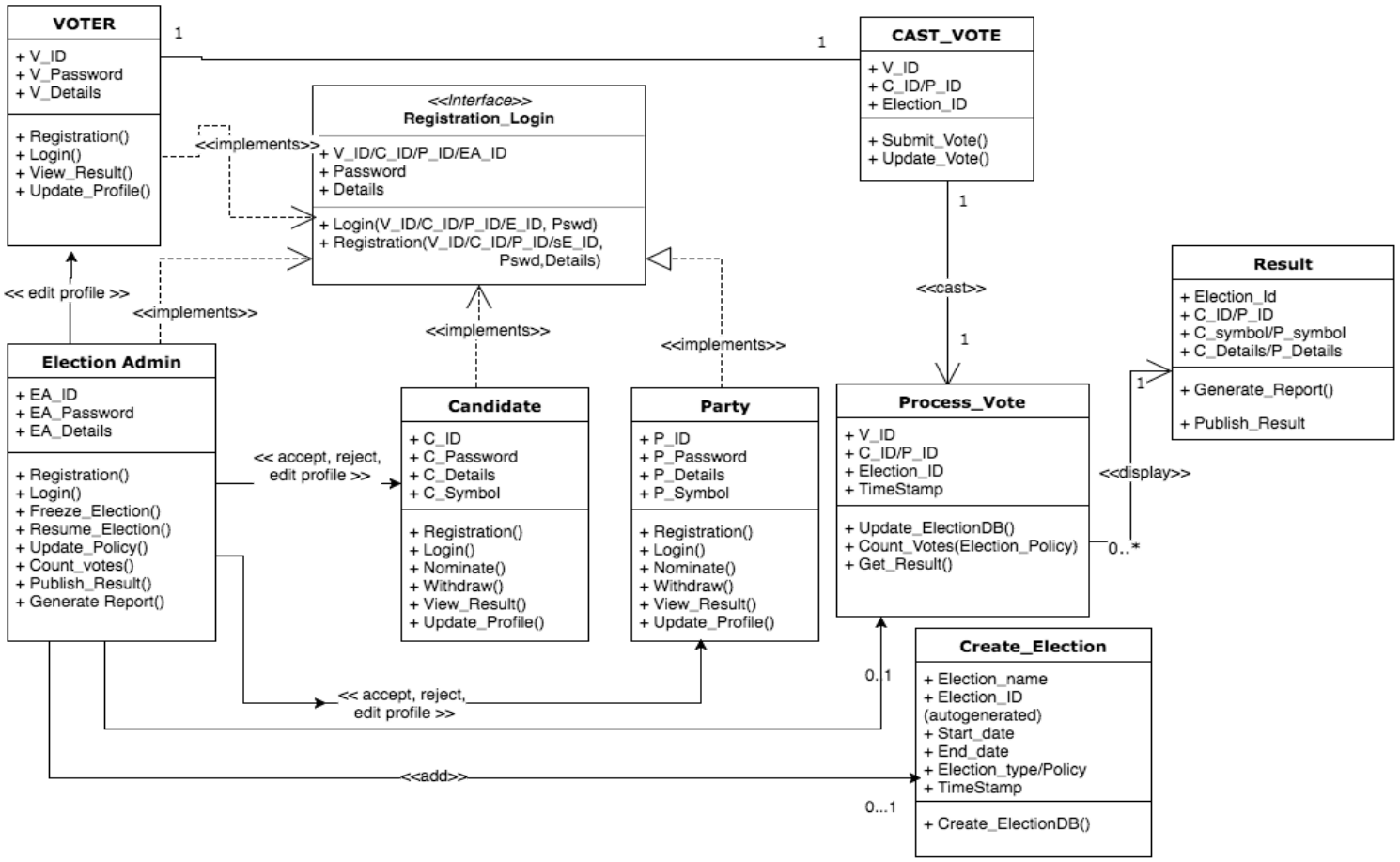
Preconditions:

- Nominee is registered

Postcondition:

- Personal details of the nominee is updated.

Domain Modelling



Design Goals

Below mentioned design goals were considered while architecting system design for Online Voting System.

Reliability
The system is reliable and designed to cater up all the basic functionalities of an Online Voting System.

Response Time
The Response Time is very minimal as we use MongoDB, Express, Angular and Node.js(MEAN stack) and the throughput rate is very high compared to other MVC frameworks.

Robustness
The system is designed to survive invalid user input. For example: if a voter wants to cast a vote for an election after the deadline, the system will notify the user that the deadline has already passed and the voting is closed.

Security
The system is secure and not vulnerable to malicious attacks. The candidate, voter details and the rules of the elections should be protected at all times and should not be accessible to any external entity.

Portability
The system should be portable. It is developed using HTML and node.js and hence can be adapted to any operating systems and platforms.

Availability
The system should be available most of the time; it can be used to accomplish normal tasks. The downtime of the server should be minimum or almost never. During downtime, new registration of the nominee will not be done. Only the registered nominees and the voters can be viewed. The system will be in a view mode only.

Usability
The system should be user-friendly so that an individual with not much basic web browsing skills can use the system with ease.

Scalability

With the increase in the number of voters and candidates for a particular election, the system should be able to handle the data accordingly and should function properly. As we have used MVC architecture to design the system, the HTML pages will get connected to the database which will handle the traffic well.

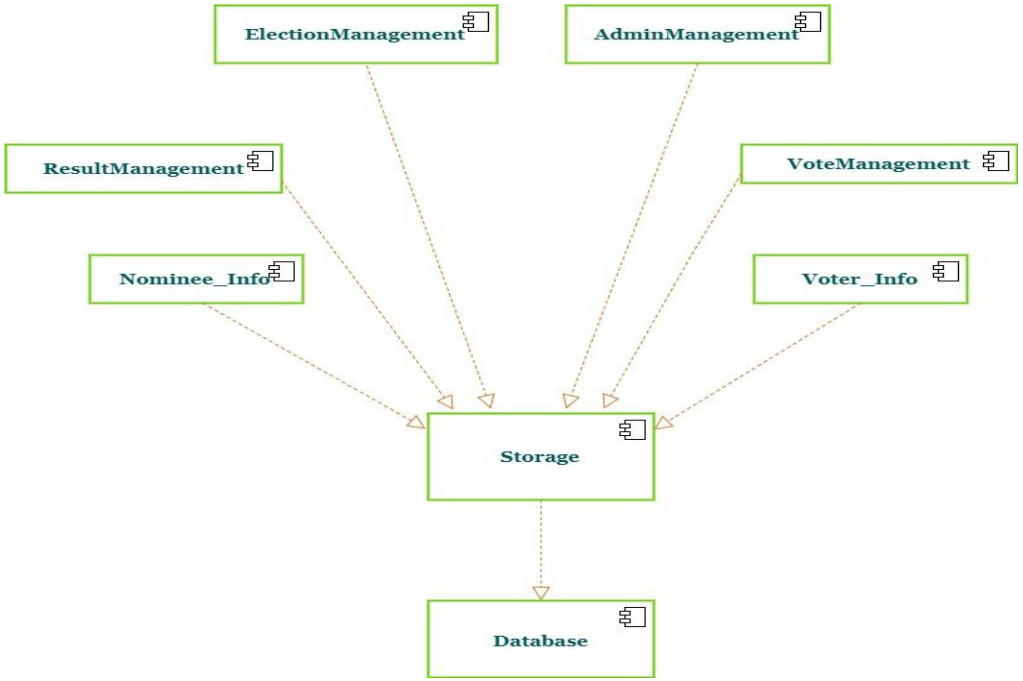
Performance

The information/data should be stored in a properly designed database. The frontend will be designed using node.js to serve the web pages and the database using MongoDB. Both the technologies are compatible with each other; hence the connectivity and the data exchange rate will be done faster. It's important to test the system thoroughly for bugs before the deployment.

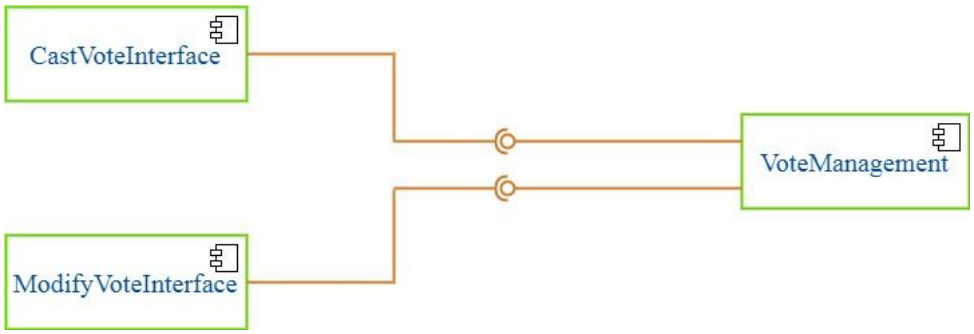
Utility

The system should support the work of Election Official and should be able to conduct the elections as per the rules the defined. The voters can cast the votes and nominees can nominate themselves. The evaluation of the results and declaring the winner should be accurate.

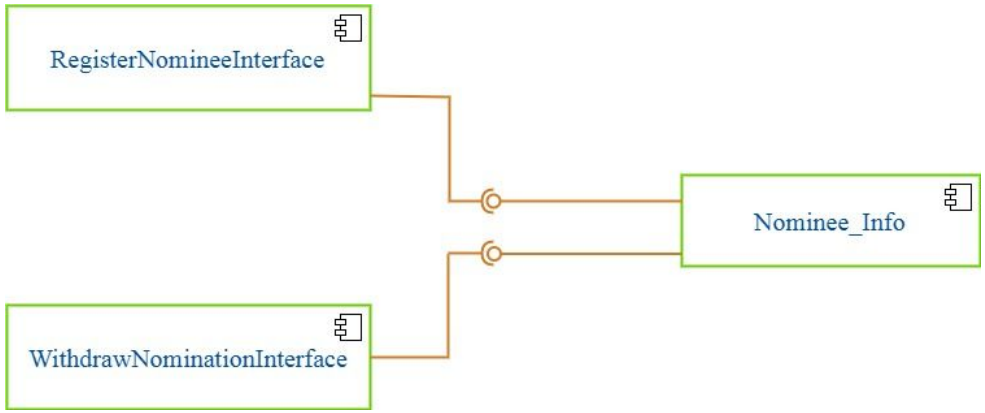
System Decomposition



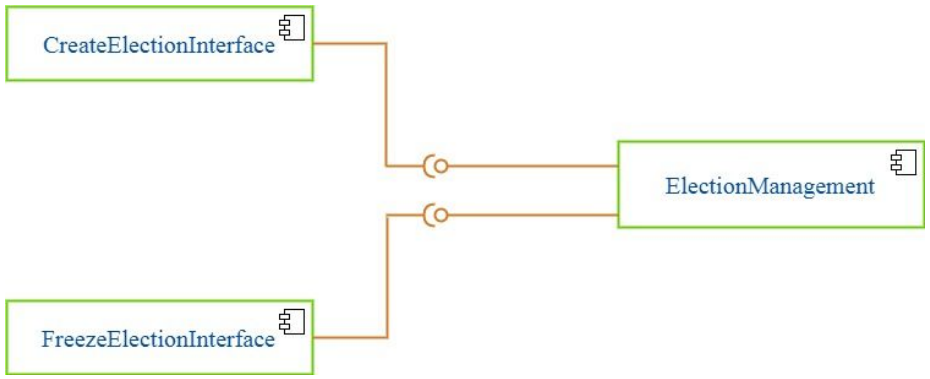
Services provided by the subsystems (UML component diagram, ball-and-socket notation depicting provided and required interfaces)



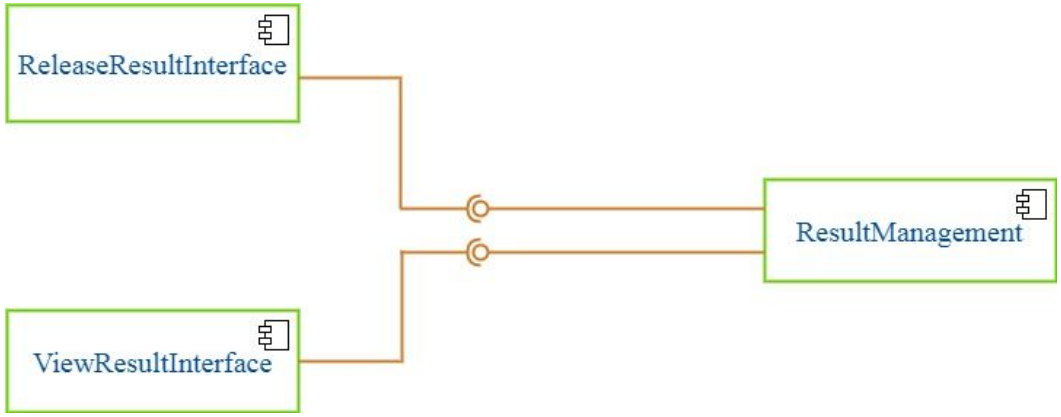
The **VoteManagement** subsystem provides services to the other modules. A voter can cast a vote by using the **CastVoteInterface** during the time elections. After the voting is done and the deadline has not passed for a particular election, **ModifyVoteInterface** enables the voter to update his vote.



RegisterNomineeInterface and **WithdrawNominationInterface** require services from the **Nominee_Info** subsystem in order to maintain the list of nominees for a particular election.



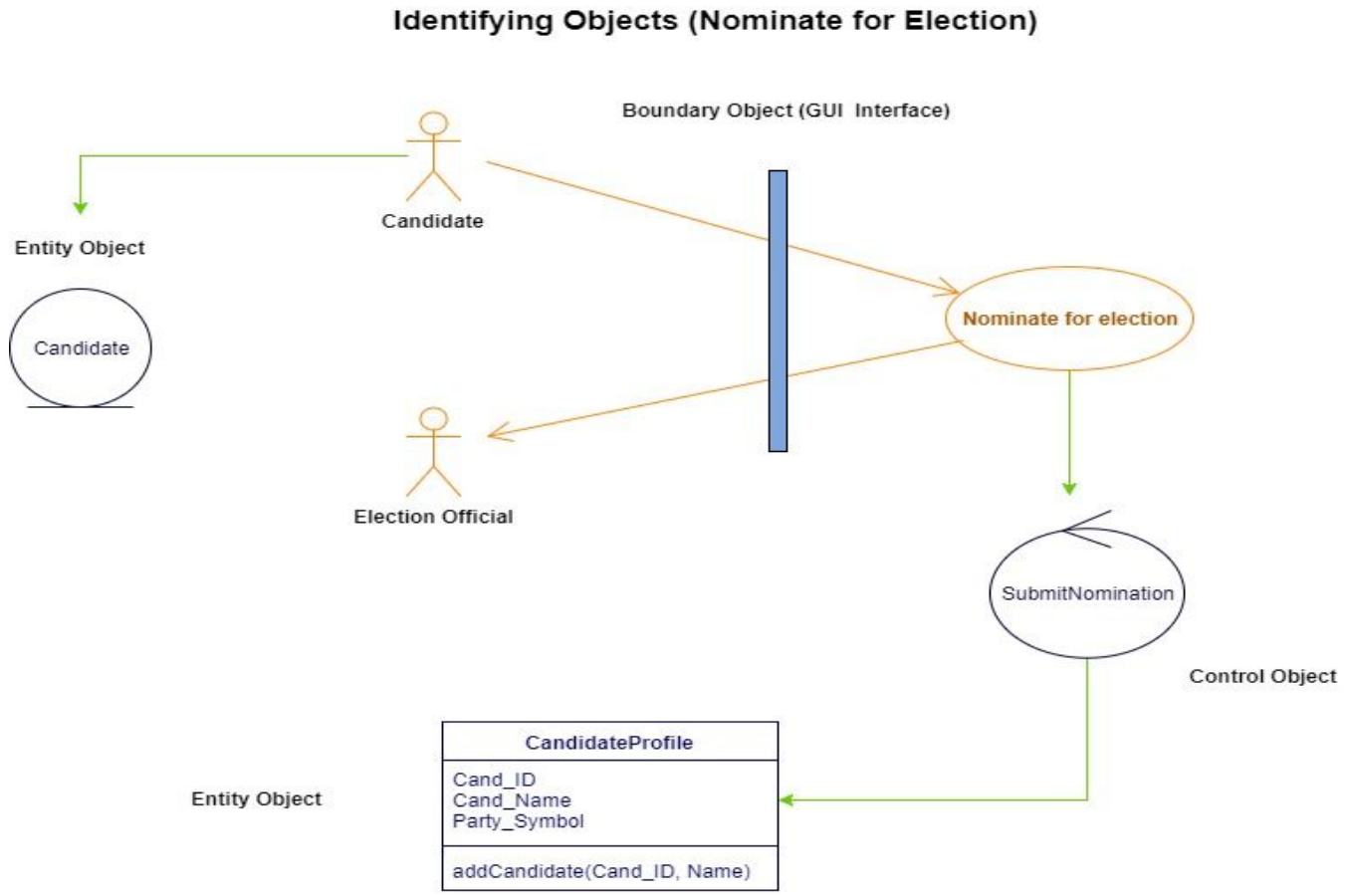
CreateElectionInterface and F**reezeElectionInterface** requires services from the **ElectionManagement** subsystem for the proper conduct of elections.



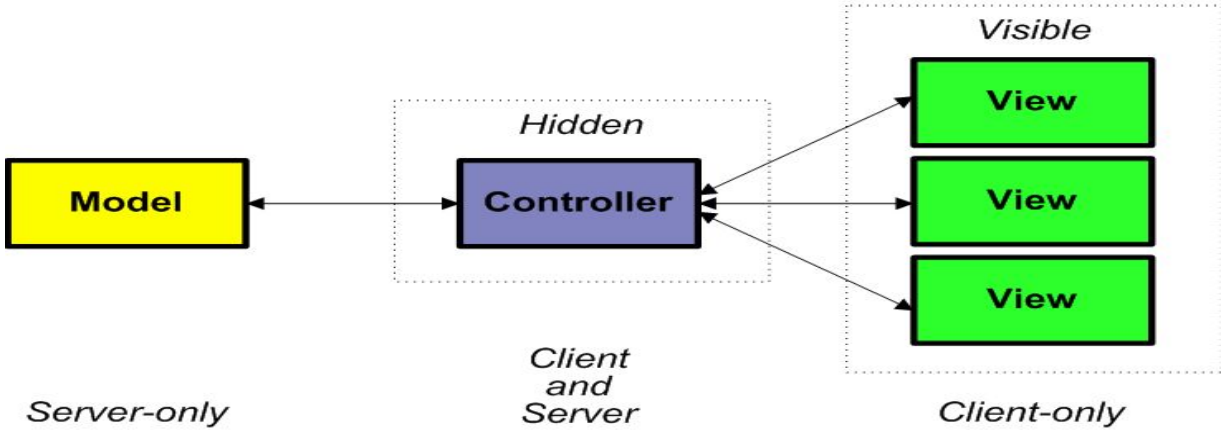
ReleaseResultInterface and **ViewResultInterface** requires services from the **ResultManagement** subsystem for declaring the winner after the elections are over.

Entity, Boundary And Control Objects

The frontend are the boundary objects, the frontend will be developed in Angular and will have MVC (Model View Controller) architecture type of it's own. So all the forms and webpages the user will see will be boundary objects. The control elements will contain in NodeJS and will have MV*(Model View) architecture type. So all the classes in domain modelling are actually controller objects. All these classes will be entities as well which will be stored in the database.



Logical Architecture: Model View Controller (MVC)

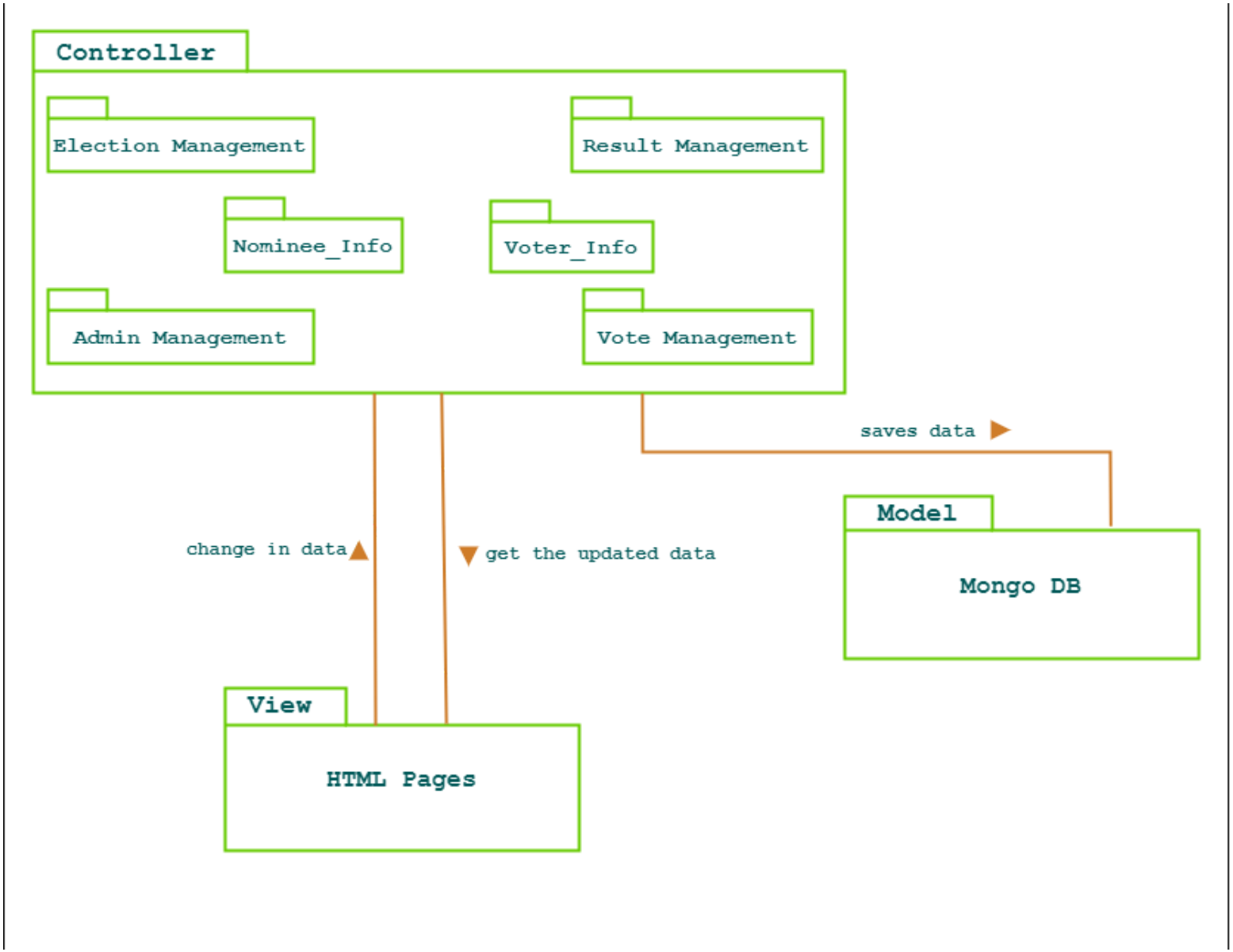


We are going to use the Model View Controller architecture for our application. Model is the central component of the architecture and it handles the access to the database server (Mongo DB). Controller acts as an intermediate and accepts the input from the user and convert it to commands for the model or view. We will be using Node.js as a controller in our application. View handles the entire HTML pages. It is the output representation of the data that the user interacts with.

Model: Mongo DB

View: HTML in Angular (MVC framework on frontend)

Controller: Node.js (MV*)


Controller
Election Management
Result Management
Nominee_Info
Voter_Info
Admin Management
Vote Management
saves data
Model
Mongo DB
change in data
get the updated data
View
HTML Pages

URLs

Website URL

http://www.cs.mun.ca/~hmk412/cs6905/a2/

Github URL

https://github.com/harisbarki/CS6905-Online-Voting